R E P U B L I Q U E F R A N Ç A I S E

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

PCT/FR 03/01058
10/509876
06 OCT 2004

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

REC'D 26 AUG 2003
WIPO PCT

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 29 JUIL 2003

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

DOCUMENT DE PRIORITE

PRÉSENTÉ OU TRANSMIS CONFORMÉMENT À LA RÈGLE 17.1.a) OU b)

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE
26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa N° 11354*01

REQUÊTE EN DÉLIVRANCE 1/2

Cet imprimé est à remplir lisiblement à l'encre noire DB 540 W /260899

Réservé à l'INPI

REMISE DES PIÈCES
DATE 3 AVRIL 2002
LIEU 54 INPI NANCY
N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0204117
DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 03 AVR. 2002

Vos références pour ce dossier *(facultatif)* 016414

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

CABINET BALLOT
9 rue Claude Chappe
Technopôle Metz 2000
57070 METZ

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	Cochez l'une des 4 cases suivantes	
Demande de brevet	☒	
Demande de certificat d'utilité	☐	
Demande divisionnaire	☐	
Demande de brevet initiale	N°	Date ___/___/___
ou demande de certificat d'utilité initiale	N°	Date ___/___/___
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N°	Date ___/___/___

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Procédé cryptographique protégé contre les attaques de type à canal caché.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	Pays ou organisation Date ___/___/___ N° Pays ou organisation Date ___/___/___ N° Pays ou organisation Date ___/___/___ N° ☐ **S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»**
5 DEMANDEUR	☐ **S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»**
Nom ou dénomination sociale	GEMPLUS
Prénoms	
Forme juridique	Société Anonyme
N° SIREN	
Code APE-NAF	
Adresse — Rue	Avenue du Pic de Bertagne Parc d'Activités de GEMENOS
Adresse — Code postal et ville	13420 GEMENOS
Pays	FRANCE
Nationalité	Française
N° de téléphone *(facultatif)*	
N° de télécopie *(facultatif)*	
Adresse électronique *(facultatif)*	

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI

REMISE DES PIÈCES
DATE 3 AVRIL 2002
LIEU 54 INPI NANCY
N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0204117

DB 540 W 7260899

Vos références pour ce dossier : *(facultatif)*		016414
6 MANDATAIRE		
Nom		LECLAIRE
Prénom		Jean-Louis
Cabinet ou Société		CABINET BALLOT
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	9 rue Claude Chappe - Technopôle Metz 2000
	Code postal et ville	57070 METZ
N° de téléphone *(facultatif)*		03 87 74 81 36
N° de télécopie *(facultatif)*		03 87 36 26 76
Adresse électronique *(facultatif)*		
7 INVENTEUR (S)		
Les inventeurs sont les demandeurs		☐ Oui ☒ Non **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 RAPPORT DE RECHERCHE		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		☒ ☐
Paiement échelonné de la redevance		**Paiement en trois versements, uniquement pour les personnes physiques** ☐ Oui ☐ Non
9 RÉDUCTION DU TAUX DES REDEVANCES		**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence) :*
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire) LECLAIRE Jean-Louis 93.4009	**VISA DE LA PRÉFECTURE OU DE L'INPI** Magali ROUX

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

PROCEDE CRYPTOGRAPHIQUE PROTEGE CONTRE LES ATTAQUES DE TYPE A CANAL CACHE

L'invention concerne un procédé cryptographique protégé contre les attaques de type à canal caché. L'invention est notamment intéressante pour protéger des algorithmes au cours duquel on exécute un bloc d'instructions parmi plusieurs bloc d'instructions différents en fonction d'une variable d'entrée. Un tel algorithme est par exemple, mais non limitativement, un algorithme binaire d'exponentiation, réalisant un calcul de type $Y = X^D$, X, Y et D étant des nombres entiers. Un tel algorithme est par exemple mis en œuvre dans des dispositifs électroniques tels que des cartes à puce.

Le diagramme de principe d'un tel algorithme est représenté sur la figure 1. Il comprend une première étape de test de la valeur d'une donnée d'entrée. En fonction du résultat du test, on réalise ensuite un bloc d'instructions Π_0 ou un bloc d'instructions Π_1. L'algorithme peut ensuite se terminer, ou bien une nouvelle étape de test est réalisée sur une autre variable d'entrée. Les blocs d'instructions Π_0, Π_1 comprennent chacun un ensemble d'instructions à exécuter. Le nombre et /ou le type des instructions peuvent être différents d'un bloc d'instructions Π_0, Π_1 à l'autre.

De nombreux algorithmes cryptographiques sont basés sur le diagramme de principe de la figure 1. C'est notamment le cas des algorithmes cryptographiques basés sur des calculs d'exponentiation du type $Y = X^D$, où X, Y sont des nombres entiers le plus souvent de grande taille, et D un nombre prédéterminé.

Les nombres X, Y peuvent correspondre par exemple à un texte chiffré ou à chiffrer, une donnée signée ou à signer, une donnée vérifiée ou à vérifier, etc. Le nombre D peut correspondre à des éléments de clés, privées ou

publiques utilisées pour le chiffrage ou le déchiffrage des nombres X, Y.

A titre d'exemple des algorithmes tels que l'algorithme dit "Square-And-Multiply", l'algorithme dit "Right-To-Left binary algorithm", l'algorithme dit "(M, M^3) algorithm" peuvent être utilisés pour réaliser des calculs d'exponentiation.

Un utilisateur malveillant peut éventuellement engager des attaques, visant à découvrir notamment des informations confidentielles (comme par exemple la clé D ou une donnée dérivée de cette clé) contenues et manipulées dans des traitements effectués par le dispositif de calcul exécutant une opération d'exponentiation.

Une attaque simple, dite "timing attack", contre l'algorithme de la figure 1 consiste à mesurer le temps nécessaire au dispositif pour exécuter un bloc d'instructions entre deux étapes de test. Si les temps d'exécution des blocs d'instructions Π_0, Π_1 sont différents, alors il est facile d'identifier un bloc d'instructions Π_0 ou Π_1 et d'en déduire la valeur de la variable d'entrée associée.

Pour se prémunir contre cette attaque, il est possible d'ajouter des instructions fictives dans le bloc d'instructions Π_0 ou Π_1 le plus court (dans le sens où le temps mis pour le réaliser est le plus petit) de sorte à ce que les deux blocs d'instructions Π_0, Π_1 soient de même durée.

Une instruction est dite fictive si son exécution ne modifie pas les données manipulées par l'algorithme. Par exemple, l'instruction i <- i - 0 est une instruction fictive.

Si cette solution est efficace contre les "timing attacks", elle n'est pas efficace contre d'autres types

d'attaques à canal caché et elle peut également pénaliser les temps d'exécution de l'algorithme.

Les attaques à canaux cachés les plus connues sont dites simples ou différentielles. On entend par attaque à canal caché une attaque basée sur une grandeur physique mesurable de l'extérieur du dispositif, et dont l'analyse directe (attaque simple) ou l'analyse selon une méthode statistique (attaque différentielle) permet de découvrir des informations contenues et manipulées dans des traitements réalisés dans le dispositif. Par exemple, dans une "timing attack", le canal caché (la grandeur physique mesurable depuis l'extérieur) est le temps.

Les attaques à canal caché peuvent permettre de découvrir des informations confidentielles. Ces attaques ont notamment été dévoilées par Paul Kocher (Advances in Cryptology - CRYPTO'99, vol. 1666 of Lecture Notes in Computer Science, pp.388-397. Springer-Verlag, 1999).

Parmi les grandeurs physiques qui peuvent être exploitées à ces fins, on peut citer le temps d'exécution, la consommation en courant, le champ électromagnétique rayonné par la partie du composant utilisée pour exécuter le calcul, etc. Ces attaques sont basées sur le fait que, au cours de l'exécution d'un algorithme, la manipulation d'un bit, c'est à dire son traitement par une instruction particulière, laisse une empreinte particulière sur la grandeur physique considérée, selon la valeur de ce bit et / ou selon l'instruction.

Les attaques à canaux cachés peuvent aboutir sur des algorithmes tels que celui de la figure 1 si les blocs d'instructions Π_0, Π_1 ne sont pas équivalents vis-à-vis de ces attaques.

Le terme "équivalent" doit être compris ici et dans tout ce qui suit de la manière suivante. Deux

instructions $INST_1$, $INST_2$ (ou deux blocs d'instructions Π_0, Π_1) sont dits équivalentes (noté $INST_0 \sim INST_1$) s'il n'est pas possible de les différencier par une attaque à canal caché. C'est le cas notamment si la grandeur physique mesurée au cours de l'attaque suit la même évolution pour les deux instructions. A noter toutefois que deux instructions peuvent être équivalentes vis-à-vis d'une attaque à canal caché et ne pas être équivalentes vis-à-vis d'une autre attaque à canal caché.

Dans le même esprit, on dira que deux instructions (ou blocs d'instructions) sont égales si lorsqu'elles sont utilisées avec les mêmes données d'entrée, elles produisent des données de sortie identiques.

Il est connu de se prémunir contre les attaques à canaux cachés en ajoutant des instructions fictives à l'algorithme. On supposera dans ce qui suit qu'une instruction fictive est équivalente à une instruction réelle similaire. Par exemple l'instruction i<-i-0 est supposée équivalente à l'instruction i<-i-1.

Dans le cas de l'algorithme de la figure 1, il est ainsi connu de réaliser un bloc d'instructions Π_1 fictif après chaque bloc d'instructions Π_0, et de réaliser de manière symétrique un bloc d'instructions Π_0 fictif avant chaque bloc d'instructions Π_1 (voir les étapes en pointillés sur la figure 1). Ainsi, quelle que soit la valeur de la donnée d'entrée, on réalisera un bloc d'instructions Π_0 et un bloc d'instructions Π_1, dans l'ordre, l'un ou l'autre étant fictif, de sorte qu'il ne soit pas possible de prédire la valeur de la donnée d'entrée, les grandeurs physiques relatives à un calcul étant équivalentes. On note ainsi :

$$(\Pi_0 \,||\, \Pi_{1(fictif)}) \sim (\Pi_{0(fictif)} \,||\, \Pi_1)$$

Si cette solution est efficace contre les attaques à canaux cachés, elle présente cependant l'inconvénient

de multiplier en moyenne par deux le temps d'exécution de l'algorithme.

En effet, dans le cas d'un algorithme non protégé utilisant M données d'entrée, on réalise statistiquement en moyenne M/2 bloc d'instructions Π_0 et M/2 bloc d'instructions Π_1. Si T0, respectivement T1 sont les temps moyens d'exécution d'un bloc d'instructions Π_0, respectivement Π_1, alors le temps moyen d'exécution de l'algorithme non protégé est égal à M*(T0 + T1) / 2.

Par contre, dans le cas de l'algorithme protégé par des blocs d'instructions Π_0, Π_1 fictifs, on réalise systématiquement un bloc d'instructions Π_0 et un bloc d'instructions Π_1 pour chacune des M données d'entrée. En conséquence, le temps moyen d'exécution de l'algorithme protégé par des blocs d'instructions fictifs est égal à M*(T0 + T1).

Un but de l'invention est de proposer un algorithme cryptographique sécurisé contre les attaques à canaux cachés et plus rapide que les algorithmes protégés existants.

Ce but est atteint par un procédé de calcul cryptographique selon l'invention, caractérisé en ce que, pour exécuter un bloc d'instructions choisi (Π_j) en fonction d'une variable d'entrée (D_i) parmi N blocs d'instructions prédéfinis (Π_1, ..., Π_N), on exécute un nombre prédéfini (L_j) de fois un bloc élémentaire commun ($\Gamma(k,s)$) aux N blocs d'instructions prédéfinis (Π_1, ..., Π_N), le nombre prédéfini (L_j) étant associé au bloc d'instructions choisi (Π_j).

Ainsi, avec l'invention on réalise un seul bloc élémentaire, le bloc élémentaire commun, un nombre prédéfini de fois, selon la variable d'entrée. Il n'est alors pas possible de déterminer par une attaque à canal caché quel bloc d'instructions est exécuté.

Le nombre prédéfini (L_j) est variable d'un bloc d'instructions prédéfini (Π_1, ..., Π_N) à l'autre.

Le bloc élémentaire commun ($\Gamma(k,s)$) comprend de préférence au moins une instruction de calcul équivalente vis-à-vis d'une attaque à canal caché à une instruction de calcul de chaque bloc prédéfini (Π_1, ..., Π_N).

Le bloc élémentaire commun ($\Gamma(k,s)$) peut également comprendre une instruction de mise à jour d'un pointeur de boucle (k) indiquant un nombre d'exécutions déjà exécutées du bloc élémentaire commun ($\Gamma(k,s)$).

Si nécessaire, le bloc élémentaire commun ($\Gamma(k,s)$) peut comprendre en complément une instruction de mise à jour d'un pointeur d'état (s) indiquant si le nombre prédéfini (L_j) a été atteint.

La valeur du pointeur de boucle (k) et / ou la valeur du pointeur d'état (s) sont fonction de la valeur de la variable d'entrée (D_i) et / ou du nombre d'instructions du bloc d'instructions (P_j) associé à la valeur de donnée d'entrée.

De préférence, si plusieurs blocs élémentaires sont possibles, le bloc élémentaire commun est choisi minimal, en ce sens qu'il comprend un nombre d'instructions minimal et / ou qu'il s'effectue en un temps minimal.

Selon un autre mode de réalisation, pour réaliser successivement plusieurs blocs d'instructions choisis parmi les N blocs d'instructions prédéfinis (Π_1, ..., Π_N) en fonction d'une variable d'entrée (D_j) associée à un indice d'entrée (i),

on exécute un nombre total (L_T) de fois le bloc élémentaire commun ($\Gamma(k,s)$), le nombre total (L_T) étant égal à une somme des nombres prédéfinis (L_j) associés à chaque bloc d'instructions choisi (Π_i).

Là encore, on exécute seulement le bloc élémentaire commun, quel que soient les blocs d'instructions à exécuter, de sorte qu'il n'est pas possible de prédire à

quel bloc d'instructions est associé le bloc élémentaire commun en cours d'exécution. Une attaque à canal caché ne peut donc pas aboutir.

A noter qu'un même bloc d'instructions peut être choisi plusieurs fois selon la variable d'entrée associée à l'indice d'entrée (i).

L'invention concerne également un procédé d'obtention d'un bloc élémentaire commun ($\Gamma(k,s)$) à N blocs d'instructions (Π_1, ..., Π_N) prédéfinis. Ledit procédé est susceptible d'être utilisé pour la mise en œuvre d'un procédé de calcul cryptographique sécurisé tel que celui décrit ci-dessus.

Selon l'invention, on obtient un bloc élémentaire commun en réalisant les étapes suivantes :

E1 : décomposition de chaque bloc d'instructions prédéfini (Π_1, ..., Π_N) en une suite de blocs élémentaires (γ) équivalents vis-à-vis d'une attaque à canal caché, et classement de l'ensemble des blocs élémentaires,

E2 : recherche d'un bloc élémentaire commun ($\Gamma(k,s)$) équivalents à tous les blocs élémentaires (γ) de tous les blocs d'instructions prédéfinis. Le bloc élémentaire commun recherché comprend au moins un bloc élémentaire précédemment obtenu lors de l'étape E1 et une instruction de mise à jour d'un pointeur de boucle (k) tel que une exécution du bloc élémentaire commun associée à la valeur du pointeur de boucle (k) et une exécution du bloc élémentaire de rang égal à la valeur du pointeur de boucle (k) soient identiques.

Si nécessaire, au cours de l'étape E1, une ou des instructions fictives peuvent être ajoutée à la série des instructions d'un ou plusieurs blocs d'instructions : ceci peut faciliter composition de chaque blocs d'instructions en des blocs élémentaires tous équivalents vis-à-vis d'une attaque à canal caché.

Au cours de l'étape E1, on découpe chaque bloc d'instructions prédéfinis Π_1 à Π_N en blocs élémentaires équivalents vis-à-vis d'une attaque donnée ; les blocs élémentaires sont classés. Par exemple :

$$\Pi_1 = \gamma 1 \,||\, \gamma 2 \,||\, \gamma 3,\ \Pi_2 = \gamma 4 \,||\, \gamma 5,\ \ldots$$

Plus généralement, chaque bloc d'instructions $\Pi_1, \ldots, \Pi_N$ se décompose ainsi :

$$\Pi_1 = \gamma(C_1) \,||\, \ldots \,||\, \gamma(C_1+L_1-1),$$
$$\Pi_2 = \gamma(C_2) \,||\, \ldots \,||\, \gamma(C_2+L_2-1),$$
$$\ldots$$
$$\Pi_j = \gamma(C_j) \,||\, \ldots \,||\, \gamma(C_j+L_j-1),$$
$$\ldots$$
$$\Pi_N = \gamma(C_N) \,||\, \ldots \,||\, \gamma(C_N+L_N-1)$$

avec
$$C_1 = 0$$
$$C_2 = L_1$$
$$\ldots$$
$$C_j = L_1+L_2+ \ldots + L_{j-1}$$
$$\ldots$$
$$C_N = L_1+ \ldots L_{N-1}$$

où L_j est le nombre de blocs élémentaires nécessaires pour décomposer complètement le bloc prédéfini d'instructions Π_j.

Au cours de l'étape E2, on recherche un bloc élémentaire commun Γ tel que chaque bloc d'instructions Π_j ($1 \leq j \leq N$) peut être exprimé sous forme d'une répétition L_j fois du bloc élémentaire commun Γ.

De préférence, le bloc commun est choisi minimal, ayant un nombre d'instructions le plus petit possible et / ou s'effectuant en un temps minimal.

Le bloc élémentaire commun peut comprendre :

- un ou plusieurs blocs élémentaires obtenus lors de l'étape E1, et

- une instruction de mise à jour d'un pointeur de boucle (k) tel que une exécution du bloc élémentaire commun associée à la valeur du pointeur de boucle (k) et

une exécution du bloc élémentaire de rang égal à la valeur du pointeur de boucle (k) soient identiques.

Si nécessaire, un pointeur d'état peut également être utilisé en complément du pointeur de boucle :

- le pointeur d'état s indique si le bloc élémentaire commun a été exécuté un nombre prédéfini de fois correspondant au nombre de blocs élémentaires décomposant un bloc d'instructions données ; dans un exemple, le pointeur d'état s est égal à 1 lorsque le nombre prédéfini de blocs élémentaires a été exécuté, et il est égal à 0 sinon.

- le pointeur de boucle indique le rang du bloc élémentaire à exécuter parmi l'ensemble des blocs élémentaires. De manière très générale, le pointeur de boucle peut être défini selon la relation suivante :

$$k <- (/s).(k+1) + s.f(D_i)$$

où D_i est la variable d'entrée permettant de sélectionner un bloc d'instructions à exécuter, s le pointeur d'état et f une fonction logique de la variable d'entrée D_i.

où f(Di) est une fonction logique de la variable d'entrée D_i associée à un bloc d'instructions prédéfini Π_j à exécuter, et /s est le complément du pointeur s (fonction NON logique).

La relation ci-dessus donnant la valeur de k est obtenue par le raisonnement suivant.

Lorsqu'on réalise un bloc d'instructions Π_j, le pointeur de boucle k doit être incrémenté de 1 à chaque exécution du bloc élémentaire commun (associée à un bloc élémentaire équivalent de la décomposition du bloc Π_j) tant que s = 0, c'est-à-dire tant que le nombre de blocs élémentaires associés au bloc Π_j n'a pas été atteint. Ceci se traduit par la relation :

$$k <- (k+1) \text{ lorsque } s = 0$$

Inversement, lorsqu'on réalise le bloc élémentaire commun associé au dernier bloc élémentaire du bloc Π_j

(c'est-à-dire lorsque s = 1), il est nécessaire d'incrémenter k de sorte à réaliser le bloc élémentaire commun associé au premier bloc élémentaire du bloc d'instructions suivant $\Pi_{j'}$. Ceci se traduit par la relation suivante :

$$k <- f(D_i) \text{ lorsque } s = 1$$

où D_i est la variable d'entrée qui conditionne le choix du calcul $\Pi_{j'}$ à réaliser.

En combinant les deux dernières relations , on obtient finalement :

$$k <- (/s).(k+1) + s.f(D_i), \text{ pour tout } s.$$

L'invention et les avantages qui en découlent apparaîtront plus clairement à la lecture de la description qui suit d'exemples de mise en œuvre d'un procédé cryptographique sécurisé selon l'invention. La description est à lire en référence aux dessins annexés dans lesquels :

- la figure 1 est un diagramme générique de procédés connus et susceptibles d'être protégés selon l'invention,
- la figure 2 est un diagramme du procédé générique de la figure 1 protégé selon l'invention,
- les figures 3 et 4 détaillent la réalisation de certaines étapes du procédé de la figure 2 dans le cas de procédés d'exponentiation connus.

Dans les exemples qui suivent, on va décrire notamment l'obtention d'un bloc élémentaire commun selon l'invention et l'utilisation de ce bloc élémentaire, dans les cas pratiques de trois procédés de calcul cryptographique.

Dans un premier exemple pratique, on considère un algorithme d'exponentiation de type "Square-and-Multiply" (qu'on appellera par la suite algorithme 1 par souci de

simplification). L'algorithme 1 permet de réaliser une opération d'exponentiation de type $Y = X^D$, où $D = (D_{M-1}, \ldots, D_0)$ est un nombre de M bits. Cet algorithme peut être représenté comme suit :

Initialisation :

$R_0 <- 1$; $R_1 <- X$; $i <- M-1$

Tant que $i \geq 0$, répéter :

Si $D_i = 1$, alors réaliser Π_0 :

$R_0 <- R_0.R_0$

$R_0 <- R_0.R_1$

$i <- i-1$

Si $D_i = 0$, alors réaliser Π_1 :

$R_0 <- R_0.R_0$

$i <- i-1$

Retourner R_0.

Algorithme "Square-and-Multiply" (algorithme 1)

Dans l'algorithme 1, les blocs d'instructions Π_0, Π_1 sont réalisés selon la valeur d'un bit de l'exposant D, et on incrémente un indice de boucle i à la fin de chaque bloc d'instructions Π_0, Π_1 de sorte à traiter successivement tous les bits de D.

Dans l'algorithme 1, les bloc d'instructions Π_0, Π_1 ne sont pas équivalents vis-à-vis d'une attaque à canal caché, notamment parce ce que le nombre d'instructions de Π_0 est différent du nombre d'instructions de Π_1.

Pour protéger l'algorithme 1 selon l'invention, on recherche un bloc élémentaire commun susceptible d'être utilisé pour l'exécution des blocs Π_0, Π_1.

Pour cela, on décompose tout d'abord chaque bloc d'instructions Π_0, Π_1 en une suite de blocs élémentaires, tous équivalents entre eux vis-à-vis d'une attaque donnée.

Le bloc d'instructions Π_0 peut s'écrire :

$R_0 <- R_0.R_0$
$i <- i-0$
$R_0 <- R_0.R_1$
$i <- i-1$

L'instruction $i <- i-0$ est fictive : elle ne modifie aucune variable, aucune donnée manipulée par l'algorithme 1.

Π_0 peut alors se décomposer en deux blocs élémentaires : $\Pi_0 = \gamma 0 \,||\, \gamma 1$ avec

$\gamma 0$: $R_0 <- R_0.R_0$
$i <- i-0$
$\gamma 1$: $R_0 <- R_0.R_1$
$i <- i-1$

Π_1 se compose en de la même façon en un bloc élémentaire :

$\Pi_1 = \gamma 2$ avec

$\gamma 2$: $R_0 <- R_0.R_0$
$i <- i-1$

On note que les blocs $\gamma 0$, $\gamma 1$, $\gamma 2$ sont tous équivalents ($\gamma 0 \sim \gamma 1 \sim \gamma 2$) si on suppose que les instructions $R_0 <- R_0.R_0$ et $R_0 <- R_0.R_1$ sont équivalentes et que les instructions $i <- i-0$ et $i <- i-1$ sont équivalentes.

On a ainsi décomposé chaque bloc d'instructions Π_0, Π_1 en un nombre variable (d'un bloc d'instructions à l'autre) de blocs élémentaires, tous équivalents les uns aux autres.

On définit ensuite un pointeur d'état s et un pointeur de rang k. Lorsqu'un bloc d'instructions Π est en cours d'exécution :

- k est utilisé pour indiquer quel bloc élémentaire γk doit être réalisé ; la valeur de k dépend notamment du bloc Π en cours d'exécution (et donc de la variable d'entrée testée)

- s est utilisé pour indiquer si au moins un bloc γk doit être encore être réalisé ou si le bloc γ en cours est le dernier du bloc Π.

Dans le cas de l'exemple ci-dessus relatif à l'algorithme 1, on peut résumer l'évolution des pointeurs k, s par le tableau ci-dessous.

		k	s
($D_i = 1$)	$\gamma 0$: $R_0 \leftarrow R_0.R_0$; $i \leftarrow i-0$	0	0
	$\gamma 1$: $R_0 \leftarrow R_0.R_1$; $i \leftarrow i-1$	1	1
($D_i = 0$)	$\gamma 2$: $R_0 \leftarrow R_0.R_0$; $i \leftarrow i-1$	2	1

tableau 1

s peut être calculé à partir de k : si le bloc élémentaire γk qui doit être réalisé est le dernier bloc élémentaire d'un bloc Π, alors s = 1, sinon s = 0.

Dans le cas de l'algorithme 1, on peut par exemple calculer s par la relation suivante (div est la division entière) :

$$s = (k \bmod 2) + (k \operatorname{div} 2)$$

On retrouve les différentes valeurs de s en fonction de k (cf tableau 1).

k peut être calculé à partir de s et de D_i, D_i étant représentatif du bloc Π en cours :

- si s = 0 (bloc Π en cours de réalisation), k est incrémenté de 1 à chaque réalisation d'un bloc élémentaire γ , pour effectuer le bloc élémentaire γ suivant.

- si s = 1, le bloc Π en cours est terminée et le bloc élémentaire suivant γ à effectuer est le premier bloc du prochain bloc Π à exécuter ; k dépend donc de D_i.

De ce qui précède, on en déduit que k peut être obtenu par la relation suivante :

$$k \leftarrow (/s).(k+1) + s.f(D_i)$$

avec /s la valeur complémentaire de s (fonction NON logique), et f une fonction logique de D_i, qui dépend de l'algorithme à protéger (voir aussi figure 3).

Dans le cas de l'algorithme 1, on peut par exemple choisir $f(D_i) = 2.(/D_i)$

Ainsi, avec la relation :

$$k \leftarrow (/s).(k+1) + s.2.(/D_i),$$

on retrouve les différentes valeurs de k en fonction de s et de D_i (cf tableau 1).

On définit finalement un bloc élémentaire commun $\Gamma(k, s)$, équivalent aux blocs élémentaires $\gamma 0$, $\gamma 1$, $\gamma 2$ et tel que $\Gamma(0, 0) = \gamma 0$, $\Gamma(1, 1) = \gamma 1$ et $\Gamma(2, 1) = \gamma 2$.

Pour l'algorithme 1, on peut par exemple choisir :

$\Gamma(k, s)$: $R_0 \leftarrow R_0.R_{k \bmod 2}$

$i \leftarrow i - s$

En utilisant la fonction Γ, l'algorithme 1 peut finalement s'écrire (voir aussi figure 3) :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$

Tant que $i \geq 0$:

répéter le bloc élémentaire commun $\Gamma(k,s)$:

$k \leftarrow (/s).(k+1) + s.2.(/D_i)$

$s \leftarrow (k \bmod 2) + (k \operatorname{div} 2)$

$R_0 \leftarrow R_0.R_{k \bmod 2}$

$i \leftarrow i - s$

Retourner R_0.

Algorithme 1 ("Square-And-Multiply") protégé

Dans cet algorithme, un seul bloc Γ est utilisé, quelles que soient les valeurs de D_i. Dans le cas où $D_i = 0$, on exécute un seul bloc Γ. Dans le cas où $D_i = 1$, on exécute successivement deux fois le bloc Γ.

Quelles que soient les valeurs des pointeurs k, s et quelle que soit la valeur de D_i, le bloc Γ associé est

équivalent, vis-à-vis d'une attaque à canal caché, au bloc précédemment exécuté et au bloc exécuté ensuite. En conséquence il n'est pas possible de les distinguer, de savoir à quel bloc d'instructions Π correspond un bloc Γ en cours.

A noter que par rapport à l'algorithme 1 non protégé, l'algorithme 1 protégé selon l'invention utilise le même nombre d'instructions de calcul (telles que des instructions de multiplication par exemple) pour aboutir au même résultat final. L'algorithme 1 protégé selon l'invention comprend simplement une étape supplémentaire de mise à jour de pointeurs : une telle étape est beaucoup plus rapide et beaucoup moins consommatrice de ressources qu'un bloc d'instructions de calcul tel qu'un bloc destiné à la réalisation d'une multiplication. En conséquence, le temps d'exécution de l'algorithme protégé est le même que celui de l'algorithme 1 non protégé : Tex = 1.5*M*T0, T0 étant le temps d'exécution d'une multiplication.

A noter également que le bloc Γ n'est pas unique pour un même algorithme.

D'autres décompositions du bloc d'instructions Π_0 peuvent être envisagées, par exemple :

$$\Pi_0 = \gamma'0 \;||\; \gamma'1 \text{ avec}$$

$$\begin{aligned} \gamma'0 : \; & R_0 \leftarrow R_0.R_0 \\ & i \leftarrow i-1 \\ \gamma'1 : \; & R_0 \leftarrow R_0.R_1 \\ & i \leftarrow i-0 \end{aligned}$$

Cette décomposition est envisageable dans la mesure où l'instruction fictive i <- i-0 peut être exécutée à tout instant au cours du bloc Π_0. On constate dès lors que les blocs élémentaires $\gamma'0$ et $\gamma2$ sont identiques. Le tableau 1 est alors modifié alors de la manière suivante.

			k	s
(D_i = 1)	$\gamma'0$: $R_0 <- R_0.R_0$;	i <- i-1	0	0
	$\gamma'1$: $R_0 <- R_0.R_1$;	i <- i-0	1	1
(D_i = 0)	$\gamma'0$: $R_0 <- R_0.R_0$;	i <- i-1	0	1

tableau 1'

Le pointeur s devient ici superflu dans la mesure où seules deux blocs élémentaires sont possibles $\gamma'0$ et $\gamma'1$. On obtient finalement le bloc élémentaire commun Γ' et l'algorithme protégé suivant (voir aussi figure 4) :

Initialisation :

$R_0 <- 1$; $R_1 <- X$; i <- M-1; k <- 1

Tant que $i \geq 0$:

répéter le bloc élémentaire commun $\Gamma'(k,s)$:

$k <- (D_i)$ ET (/k)

$R_0 <- R_0.R_k$

i <- i - (/k)

Retourner R_0.

Algorithme "Square-And-Multiply" protégé (2ème version)

D'autres algorithmes peuvent également être protégés selon l'invention.

L'algorithme d'exponentiation dit "Right-To-Left binary algorithm" (appelé par la suite algorithme 2) par exemple est assez similaire à l'algorithme "Square-And Multiply". Il permet de réaliser une opération de type $Y = X^D$ en partant du bit le plus faible de D de la manière suivante :

Initialisation :

$R_0 <- 1$; $R_1 <- X$; i <- 0

Tant que $i \leq M-1$, répéter :

Si $D_i = 1$, alors réaliser le bloc Π_0 :

$R_0 <- R_0.R_1$
$R_1 <- R_1.R_1$
$i <- i+1$

Si $D_i = 0$, alors réaliser le bloc Π_1 :
$R_1 <- R_1.R_1$
$i <- i+1$

Retourner R_0.

Algorithme 2, dit "Right-To-Left binary algorithm"

Les bloc Π_0, Π_1 peuvent par exemple se décomposer de la manière suivante :

		k	s
Π_0 ($D_i = 1$)	$\gamma 0$: $R_0 <- R_0.R_1$; $i <- i+0$	0	0
	$\gamma 1$: $R_1 <- R_1.R_1$; $i <- i+1$	1	1
Π_1 ($D_i = 0$)	$\gamma 2$: $R_1 <- R_1.R_1$; $i <- i+1$	2	1

tableau 2

Ici également, comme seuls deux blocs élémentaires $\gamma 0$, $\gamma 1$ sont utilisés pour décomposer Π_0, Π_1, le pointeur s est donc inutile. On peut par exemple choisir le bloc élémentaire commun Γ suivant :

$\Gamma(k)$: $R_k <- R_k.R_1$
$i <- i+k$

et mettre à jour le pointeur k avant chaque réalisation du bloc $\Gamma(k)$ selon l'instruction $k <- k \oplus D_i$. On obtient finalement l'algorithme 2 protégé ci-dessous, où $\oplus$ désigne l'opérateur OU-Exclusif :

Initialisation :
$R_0 <- 1$; $R_1 <- X$; $i <- 0$; $k <- 1$
Tant que $i \leq M-1$:
répéter le bloc élémentaire commun $\Gamma(k,s)$:
$k <- k \oplus D_i$

R_k <- $R_k.R_1$
i <- i+k
Retourner R_0.

Algorithme "Right-To-Left binary algorithm" (algorithme 2) protégé

Les exemples ci-dessus décrivent des algorithmes au cours desquels on exécute deux blocs d'instructions Π_0 ou Π_1 en fonction d'une variable d'entrée D_i. L'invention peut cependant s'appliquer à des algorithmes utilisant plus de deux blocs d'instructions Π. Par exemple, si on considère l'algorithme dit "(M, M^3) algorithm" (appelé par la suite algorithme 3) :

Initialisation :
R_0 <- 1 ; R_1 <- X ; R_2 <- X^3 ; D_{-1} <- 0 ; i <- M-1
Tant que i ≥ 0, répéter :
Si D_i = 0, réaliser le bloc d'instructions Π_0 :
R_0 <- $(R_0)^2$
i <- i-1
Si D_i = 1 ET (D_{i-1} = 0), réaliser le bloc Π_1 :
R_0 <- $(R_0)^2$
R_0 <- $R_0.R_1$
i <- i-1
Si D_i = 1 ET (D_{i-1} = 1), réaliser le bloc Π_2 :
R_0 <- $(R_0)^2$
R_0 <- $(R_0)^2$
R_0 <- $R_0.R_2$
i <- i-2
Retourner R_0.

Algorithme dit " (M, M^3) algorithm" (algorithme 3)

ET est la fonction ET logique.

En remplaçant les carrés $(R_0)^2$ par des multiplications $R_0.R_0$, et en introduisant des instructions

fictives de type i <- i-0, on peut décomposer l'algorithme 3 selon le tableau :

				k	s
Π_0	$(D_i = 0)$	$\gamma 0$	: $R_0 <- R_0.R_0$; i <- i-1	0	1
Π_1	$(D_i = 1)$ et $(D_{i-1} = 0)$	$\gamma 1$	: $R_0 <- R_0.R_0$; i <- i-0	1	0
		$\gamma 2$	: $R_0 <- R_0.R_1$; i <- i-1	2	1
Π_2	$(D_i = 1)$ et $(D_{i-1} = 1)$	$\gamma 3$	: $R_0 <- R_0.R_0$; i <- i-0	3	0
		$\gamma 4$	: $R_0 <- R_0.R_0$; i <- i-0	4	0
		$\gamma 5$	: $R_0 <- R_0.R_2$; i <- i-2	5	1

tableau 3

Le tableau 3 permet de calculer assez aisément le pointeur k en fonction de s et de D_i, et le pointeur s en fonction de k, comme précédemment. Par ailleurs, les blocs $\gamma 0$ à $\gamma 5$ sont tous équivalents, et on peut par exemple choisir le bloc élémentaire commun $\Gamma(k, s)$ suivant :

$\Gamma(k, s)$: $R_0 <- R_0.R_{s.(k \text{ div } 2)}$
i <- i - s.(k mod 2 + 1)

On en déduit finalement un algorithme 3 protégé :

Initialisation :
$R_0 <- 1$; $R_1 <- X$; $R_2 <- X^3$;
$D_{-1} <- 0$; i <- M-1 ; s <- 1
Tant que $i \geq 0$:
répéter le bloc élémentaire $\Gamma(k,s)$:
k <- (/s).(k+1) + s.(D_i + 2.(D_i ET D_{i-1}))
s <- /((k mod 2) $\oplus$ (k div 4))
$R_0 <- R_0.R_{s.(k \text{ div } 2)}$
i <- i - s.(k mod 2 + 1)
Retourner R_0.

Algorithme dit " (M, M^3) algorithm"
(algorithme 3) protégé

REVENDICATIONS

1. Procédé de calcul cryptographique caractérisé en ce que, pour exécuter un bloc d'instructions choisi (Π_j) en fonction d'une variable d'entrée (D_i) parmi N blocs d'instructions prédéfinis (Π_1, ..., Π_N), on exécute un nombre prédéfini (L_j) de fois un bloc élémentaire commun ($\Gamma(k,s)$) aux N blocs d'instructions prédéfinis (Π_1, ..., Π_N), le nombre prédéfini (L_j) étant associé au bloc d'instructions choisi (Π_j).

2. Procédé selon la revendication 1, dans lequel le nombre prédéfini (L_j) est variable d'un bloc d'instructions prédéfini (Π_1, ..., Π_N) à l'autre.

3. Procédé selon l'une des revendications 1 à 2, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend au moins une instruction de calcul équivalente vis-à-vis d'une attaque à canal caché à une instruction de calcul de chaque bloc prédéfini (Π_1, ..., Π_N).

4. Procédé selon la revendication 3, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur de boucle (k) indiquant un nombre d'exécutions déjà exécutées du bloc élémentaire commun ($\Gamma(k,s)$).

5. Procédé selon la revendication 3 ou la revendication 4, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur d'état (s) indiquant si le nombre prédéfini (L_j) a été atteint.

6. Procédé selon la revendication 4 ou la revendication 5, dans lequel la valeur du pointeur de

REVENDICATIONS

1. Procédé de calcul cryptographique caractérisé en ce que, pour exécuter un bloc d'instructions choisi (Π_j) en fonction d'une variable d'entrée (D_i) parmi N blocs d'instructions prédéfinis (Π_1, ..., Π_N), on exécute un nombre prédéfini (L_j) de fois un bloc élémentaire commun ($\Gamma(k,s)$) aux N blocs d'instructions prédéfinis (Π_1, ..., Π_N), le nombre prédéfini (L_j) étant associé au bloc d'instructions choisi (Π_j).

2. Procédé selon la revendication 1, dans lequel le nombre prédéfini (L_j) est variable d'un bloc d'instructions prédéfini (Π_1, ..., Π_N) à l'autre.

3. Procédé selon l'une des revendications 1 à 2, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend au moins une instruction de calcul équivalente vis-à-vis d'une attaque à canal caché à une instruction de calcul de chaque bloc prédéfini (Π_1, ..., Π_N).

4. Procédé selon la revendication 3, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur de boucle (k) indiquant un nombre d'exécutions déjà exécutées du bloc élémentaire commun ($\Gamma(k,s)$).

5. Procédé selon la revendication 3 ou la revendication 4, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur d'état (s) indiquant si le nombre prédéfini (L_j) a été atteint.

6. Procédé selon la revendication 4 ou la revendication 5, dans lequel la valeur du pointeur de

REVENDICATIONS

1. Procédé de calcul cryptographique caractérisé en ce que, pour exécuter un bloc d'instructions choisi (Π_j) en fonction d'une variable d'entrée (D_i) parmi N blocs d'instructions prédéfinis (Π_1, ..., Π_N), on exécute un nombre prédéfini (L_j) de fois un bloc élémentaire commun ($\Gamma(k,s)$) aux N blocs d'instructions prédéfinis (Π_1, ..., Π_N), le nombre prédéfini (L_j) étant associé au bloc d'instructions choisi (Π_j).

2. Procédé selon la revendication 1, dans lequel le nombre prédéfini (L_j) est variable d'un bloc d'instructions prédéfini (Π_1, ..., Π_N) à l'autre.

3. Procédé selon l'une des revendications 1 à 2, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend au moins une instruction de calcul équivalente vis-à-vis d'une attaque à canal caché à une instruction de calcul de chaque bloc prédéfini (Π_1, ..., Π_N).

4. Procédé selon la revendication 3, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur de boucle (k) indiquant un nombre d'exécutions déjà exécutées du bloc élémentaire commun ($\Gamma(k,s)$).

5. Procédé selon la revendication 3 ou la revendication 4, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur d'état (s) indiquant si le nombre prédéfini (L_j) a été atteint.

6. Procédé selon la revendication 4 ou la revendication 5, dans lequel la valeur du pointeur de

REVENDICATIONS

1. Procédé de calcul cryptographique caractérisé en ce que, pour exécuter un bloc d'instructions choisi (Π_j) en fonction d'une variable d'entrée (D_i) parmi N blocs d'instructions prédéfinis (Π_1, ..., Π_N), on exécute un nombre prédéfini (L_j) de fois un bloc élémentaire commun ($\Gamma(k,s)$) aux N blocs d'instructions prédéfinis (Π_1, ..., Π_N), le nombre prédéfini (L_j) étant associé au bloc d'instructions choisi (Π_j).

2. Procédé selon la revendication 1, dans lequel le nombre prédéfini (L_j) est variable d'un bloc d'instructions prédéfini (Π_1, ..., Π_N) à l'autre.

3. Procédé selon l'une des revendications 1 à 2, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend au moins une instruction de calcul équivalente vis-à-vis d'une attaque à canal caché à une instruction de calcul de chaque bloc prédéfini (Π_1, ..., Π_N).

4. Procédé selon la revendication 3, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur de boucle (k) indiquant un nombre d'exécutions déjà exécutées du bloc élémentaire commun ($\Gamma(k,s)$).

5. Procédé selon la revendication 3 ou la revendication 4, dans lequel le bloc élémentaire commun ($\Gamma(k,s)$) comprend également une instruction de mise à jour d'un pointeur d'état (s) indiquant si le nombre prédéfini (L_j) a été atteint.

6. Procédé selon la revendication 4 ou la revendication 5, dans lequel la valeur du pointeur de

boucle (k) et ou la valeur du pointeur d'état (s) sont fonction de la valeur de la variable d'entrée (D_i) et / ou du nombre d'instructions du bloc d'instructions (P_j) associé à la valeur de donnée d'entrée.

7. Procédé selon l'une des revendications 1 à 6, dans lequel, pour réaliser successivement plusieurs blocs d'instructions choisis parmi les N blocs d'instructions prédéfinis (Π_1, ..., Π_N) en fonction d'une variable d'entrée (D_j) associée à un indice d'entrée (i),

on exécute un nombre total (L_T) de fois le bloc élémentaire commun ($\Gamma(k,s)$), le nombre total (L_T) étant égal à une somme des nombres prédéfinis (L_j) associés à chaque bloc d'instructions choisi (Π_i).

8. Procédé selon la revendication 7, au cours duquel un même bloc d'instructions peut être choisi plusieurs fois selon la variable d'entrée associée à l'indice d'entrée (i).

9. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$

Tant que $i \geq 0$:

répéter le bloc élémentaire $\Gamma(k,s)$:

$k \leftarrow (/s).(k+1) + s.2.(/D_i)$

$s \leftarrow (k \bmod 2) + (k \operatorname{div} 2)$

$R_0 \leftarrow R_0.R_{k \bmod 2}$

$i \leftarrow i - s$

Retourner R_0.

boucle (k) et ou la valeur du pointeur d'état (s) sont fonction de la valeur de la variable d'entrée (D_i) et / ou du nombre d'instructions du bloc d'instructions (P_j) associé à la valeur de donnée d'entrée.

7. Procédé selon l'une des revendications 1 à 6, dans lequel, pour réaliser successivement plusieurs blocs d'instructions choisis parmi les N blocs d'instructions prédéfinis (Π_1, ..., Π_N) en fonction d'une variable d'entrée (D_j) associée à un indice d'entrée (i),

on exécute un nombre total (L_T) de fois le bloc élémentaire commun ($\Gamma(k,s)$), le nombre total (L_T) étant égal à une somme des nombres prédéfinis (L_j) associés à chaque bloc d'instructions choisi (Π_i).

8. Procédé selon la revendication 7, au cours duquel un même bloc d'instructions peut être choisi plusieurs fois selon la variable d'entrée associée à l'indice d'entrée (i).

9. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$

Tant que $i \geq 0$:

répéter le bloc élémentaire $\Gamma(k,s)$:

$k \leftarrow (/s).(k+1) + s.2.(/D_i)$

$s \leftarrow (k \bmod 2) + (k \operatorname{div} 2)$

$R_0 \leftarrow R_0.R_{k \bmod 2}$

$i \leftarrow i - s$

Retourner R_0.

boucle (k) et ou la valeur du pointeur d'état (s) sont fonction de la valeur de la variable d'entrée (D_i) et / ou du nombre d'instructions du bloc d'instructions (P_j) associé à la valeur de donnée d'entrée.

7. Procédé selon l'une des revendications 1 à 6, dans lequel, pour réaliser successivement plusieurs blocs d'instructions choisis parmi les N blocs d'instructions prédéfinis (Π_1, ..., Π_N) en fonction d'une variable d'entrée (D_j) associée à un indice d'entrée (i),

on exécute un nombre total (L_T) de fois le bloc élémentaire commun ($\Gamma(k,s)$), le nombre total (L_T) étant égal à une somme des nombres prédéfinis (L_j) associés à chaque bloc d'instructions choisi (Π_i).

8. Procédé selon la revendication 7, au cours duquel un même bloc d'instructions peut être choisi plusieurs fois selon la variable d'entrée associée à l'indice d'entrée (i).

9. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$

Tant que $i \geq 0$:

répéter le bloc élémentaire $\Gamma(k,s)$:

$k \leftarrow (/s).(k+1) + s.2.(/D_i)$

$s \leftarrow (k \bmod 2) + (k \operatorname{div} 2)$

$R_0 \leftarrow R_0.R_{k \bmod 2}$

$i \leftarrow i - s$

Retourner R_0.

boucle (k) et ou la valeur du pointeur d'état (s) sont fonction de la valeur de la variable d'entrée (D_i) et / ou du nombre d'instructions du bloc d'instructions (P_j) associé à la valeur de donnée d'entrée.

7. Procédé selon l'une des revendications 1 à 6, dans lequel, pour réaliser successivement plusieurs blocs d'instructions choisis parmi les N blocs d'instructions prédéfinis (Π_1, ..., Π_N) en fonction d'une variable d'entrée (D_j) associée à un indice d'entrée (i),

on exécute un nombre total (L_T) de fois le bloc élémentaire commun ($\Gamma(k,s)$), le nombre total (L_T) étant égal à une somme des nombres prédéfinis (L_j) associés à chaque bloc d'instructions choisi (Π_i).

8. Procédé selon la revendication 7, au cours duquel un même bloc d'instructions peut être choisi plusieurs fois selon la variable d'entrée associée à l'indice d'entrée (i).

9. Procédé selon l'une des revendications 7 ou 8, utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$

Tant que $i \geq 0$:

répéter le bloc élémentaire $\Gamma(k,s)$:

$k \leftarrow (/s).(k+1) + s.2.(/D_i)$

$s \leftarrow (k \bmod 2) + (k \operatorname{div} 2)$

$R_0 \leftarrow R_0.R_{k \bmod 2}$

$i \leftarrow i - s$

Retourner R_0.

10. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$; $k \leftarrow 1$

Tant que $i \geq 0$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k \leftarrow (D_i)$ ET $(/k)$

$R_0 \leftarrow R_0.R_k$

$i \leftarrow i - (/k)$

Retourner R_0.

11. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow 0$; $k \leftarrow 1$

Tant que $i \leq M-1$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k \leftarrow k \oplus D_i$

$R_k \leftarrow R_k.R_1$

$i \leftarrow i+k$

Retourner R_0.

12. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

10. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$; $k \leftarrow 1$

Tant que $i \geq 0$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k \leftarrow (D_i)$ ET $(/k)$

$R_0 \leftarrow R_0.R_k$

$i \leftarrow i - (/k)$

Retourner R_0.

11. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow 0$; $k \leftarrow 1$

Tant que $i \leq M-1$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k \leftarrow k \oplus D_i$

$R_k \leftarrow R_k.R_1$

$i \leftarrow i+k$

Retourner R_0.

12. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

10. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow M-1$; $k \leftarrow 1$

Tant que $i \geq 0$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k \leftarrow (D_i)$ ET $(/k)$

$R_0 \leftarrow R_0.R_k$

$i \leftarrow i - (/k)$

Retourner R_0.

11. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 \leftarrow 1$; $R_1 \leftarrow X$; $i \leftarrow 0$; $k \leftarrow 1$

Tant que $i \leq M-1$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k \leftarrow k \oplus D_i$

$R_k \leftarrow R_k.R_1$

$i \leftarrow i+k$

Retourner R_0.

12. Procédé selon l'une des revendications 7 ou 8, susceptible d'être utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

10. Procédé selon l'une des revendications 7 ou 8, utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 <- 1$; $R_1 <- X$; $i <- M-1$; $k <- 1$

Tant que $i \geq 0$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k <- (D_i)$ ET $(/k)$

$R_0 <- R_0.R_k$

$i <- i - (/k)$

Retourner R_0.

11. Procédé selon l'une des revendications 7 ou 8, utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

$R_0 <- 1$; $R_1 <- X$; $i <- 0$; $k <- 1$

Tant que $i \leq M-1$:

répéter le bloc élémentaire commun $\Gamma(k)$:

$k <- k \oplus D_i$

$R_k <- R_k.R_1$

$i <- i+k$

Retourner R_0.

12. Procédé selon l'une des revendications 7 ou 8, utilisé dans la mise en œuvre d'un calcul d'exponentiation de type $Y = X^D$, D étant un nombre entier de M bits, chaque bit (D_i) correspondant à une variable d'entrée d'indice d'entrée i, le procédé comprenant les étapes suivantes :

Initialisation :

R_0 <- 1 ; R_1 <- X ; R_2 <- X^3 ;

D_{-1} <- 0 ; i <- M-1 ; s <- 1

Tant que i ≥ 0 :

répéter le bloc élémentaire commun $\Gamma(k,s)$:

k <- (/s).(k+1) + s.(D_i + 2.(D_i ET D_{i-1}))

s <- /((k mod 2) ⊕ (k div 4))

R_0 <- $R_0.R_{s.(k \text{ div } 2)}$

i <- i - s.(k mod 2 + 1)

Retourner R_0.

13. Procédé d'obtention d'un bloc élémentaire commun ($\Gamma(k,s)$) à N blocs d'instructions (Π_1, ..., Π_N) prédéfinis, le procédé étant caractérisé en ce qu'il comprend les étapes suivantes :

E1 : décomposition de chaque bloc d'instructions prédéfini (Π_1, ..., Π_N) en une suite de blocs élémentaires (γ) équivalents vis-à-vis d'une attaque à canal caché, et classement de l'ensemble des blocs élémentaires,

E2 : recherche d'un bloc élémentaire commun ($\Gamma(k,s)$) équivalents à tous les blocs élémentaires (γ) de tous les blocs d'instructions prédéfinis, le bloc élémentaire commun comprenant au moins un bloc élémentaire précédemment obtenu et une instruction de mise à jour d'un pointeur de boucle (k) tel que une exécution du bloc élémentaire commun associée à la valeur du pointeur de boucle (k) et une exécution du bloc élémentaire de rang égal à la valeur du pointeur de boucle (k) soient identiques.

14. Procédé selon la revendication 13, caractérisé en ce que au cours de l'étape E1, on ajoute au moins une instruction fictive à au moins un bloc d'instructions prédéfini.

Initialisation :

R_0 <- 1 ; R_1 <- X ; R_2 <- X^3 ;

D_{-1} <- 0 ; i <- M-1 ; s <- 1

Tant que i ≥ 0 :

répéter le bloc élémentaire commun $\Gamma(k,s)$:

k <- (/s).(k+1) + s.(D_i + 2.(D_i ET D_{i-1}))

s <- /((k mod 2) ⊕ (k div 4))

R_0 <- $R_0.R_{a.(k \text{ div } 2)}$

i <- i - s.(k mod 2 + 1)

Retourner R_0.

13. Procédé d'obtention d'un bloc élémentaire commun ($\Gamma(k,s)$) à N blocs d'instructions (Π_1, ..., Π_N) prédéfinis, procédé susceptible d'être utilisé pour la mise en œuvre d'un procédé de calcul cryptographique selon l'une des revendications 1 à 12, le procédé étant caractérisé en ce qu'il comprend les étapes suivantes :

E1 : décomposition de chaque bloc d'instructions prédéfini (Π_1, ..., Π_N) en une suite de blocs élémentaires (γ) équivalents vis-à-vis d'une attaque à canal caché, et classement de l'ensemble des blocs élémentaires,

E2 : recherche d'un bloc élémentaire commun ($\Gamma(k,s)$) équivalents à tous les blocs élémentaires (γ) de tous les blocs d'instructions prédéfinis, le bloc élémentaire commun comprenant au moins un bloc élémentaire précédemment obtenu et une instruction de mise à jour d'un pointeur de boucle (k) tel que une exécution du bloc élémentaire commun associée à la valeur du pointeur de boucle (k) et une exécution du bloc élémentaire de rang égal à la valeur du pointeur de boucle (k) soient identiques.

14. Procédé selon la revendication 13, caractérisé en ce que au cours de l'étape E1, on ajoute au moins une

Initialisation :

R_0 <- 1 ; R_1 <- X ; R_2 <- X^3 ;

D_{-1} <- 0 ; i <- M-1 ; s <- 1

Tant que i ≥ 0 :

répéter le bloc élémentaire commun $\Gamma(k,s)$:

~~k <- (/s).(k+1) + s.(D_i + 2.(D_i ET D_{i-1}))~~

s <- /((k mod 2) ⊕ (k div 4))

R_0 <- $R_0.R_{s.(k \text{ div } 2)}$

i <- i - s.(k mod 2 + 1)

Retourner R_0.

13. Procédé d'obtention d'un bloc élémentaire commun ($\Gamma(k,s)$) à N blocs d'instructions (Π_1, ..., Π_N) prédéfinis, procédé susceptible d'être utilisé pour la mise en œuvre d'un procédé de calcul cryptographique selon l'une des revendications 1 à 12, le procédé étant caractérisé en ce qu'il comprend les étapes suivantes :

E1 : décomposition de chaque bloc d'instructions prédéfini (Π_1, ..., Π_N) en une suite de blocs élémentaires (γ) équivalents vis-à-vis d'une attaque à canal caché, et classement de l'ensemble des blocs élémentaires,

E2 : recherche d'un bloc élémentaire commun ($\Gamma(k,s)$) équivalents à tous les blocs élémentaires (γ) de tous les blocs d'instructions prédéfinis, le bloc élémentaire commun comprenant au moins un bloc élémentaire précédemment obtenu et une instruction de mise à jour d'un pointeur de boucle (k) tel que une exécution du bloc élémentaire commun associée à la valeur du pointeur de boucle (k) et une exécution du bloc élémentaire de rang égal à la valeur du pointeur de boucle (k) soient identiques.

14. Procédé selon la revendication 13, caractérisé en ce que au cours de l'étape E1, on ajoute au moins une

Initialisation :

R_0 <- 1 ; R_1 <- X ; R_2 <- X^3 ;

D_{-1} <- 0 ; i <- M-1 ; s <- 1

Tant que i ≥ 0 :

répéter le bloc élémentaire commun $\Gamma(k,s)$:

k <- (/s).(k+1) + s.(D_i + 2.(D_i ET D_{i-1}))

s <- /((k mod 2) ⊕ (k div 4))

R_0 <- $R_0.R_{s.(k \text{ div } 2)}$

i <- i - s.(k mod 2 + 1)

Retourner R_0.

13. Procédé d'obtention d'un bloc élémentaire commun ($\Gamma(k,s)$) à N blocs d'instructions (Π_1, ..., Π_N) prédéfinis, procédé susceptible d'être utilisé pour la mise en œuvre d'un procédé de calcul cryptographique selon l'une des revendications 1 à 12, le procédé étant caractérisé en ce qu'il comprend les étapes suivantes :

E1 : décomposition de chaque bloc d'instructions prédéfini (Π_1, ..., Π_N) en une suite de blocs élémentaires (γ) équivalents vis-à-vis d'une attaque à canal caché, et classement de l'ensemble des blocs élémentaires,

E2 : recherche d'un bloc élémentaire commun ($\Gamma(k,s)$) équivalents à tous les blocs élémentaires (γ) de tous les blocs d'instructions prédéfinis, le bloc élémentaire commun comprenant au moins un bloc élémentaire précédemment obtenu et une instruction de mise à jour d'un pointeur de boucle (k) tel que une exécution du bloc élémentaire commun associée à la valeur du pointeur de boucle (k) et une exécution du bloc élémentaire de rang égal à la valeur du pointeur de boucle (k) soient identiques.

14. Procédé selon la revendication 13, caractérisé en ce que au cours de l'étape E1, on ajoute au moins une instruction fictive à au moins un bloc d'instructions prédéfini.

instruction fictive à au moins un bloc d'instructions prédéfini.

instruction fictive à au moins un bloc d'instructions prédéfini.

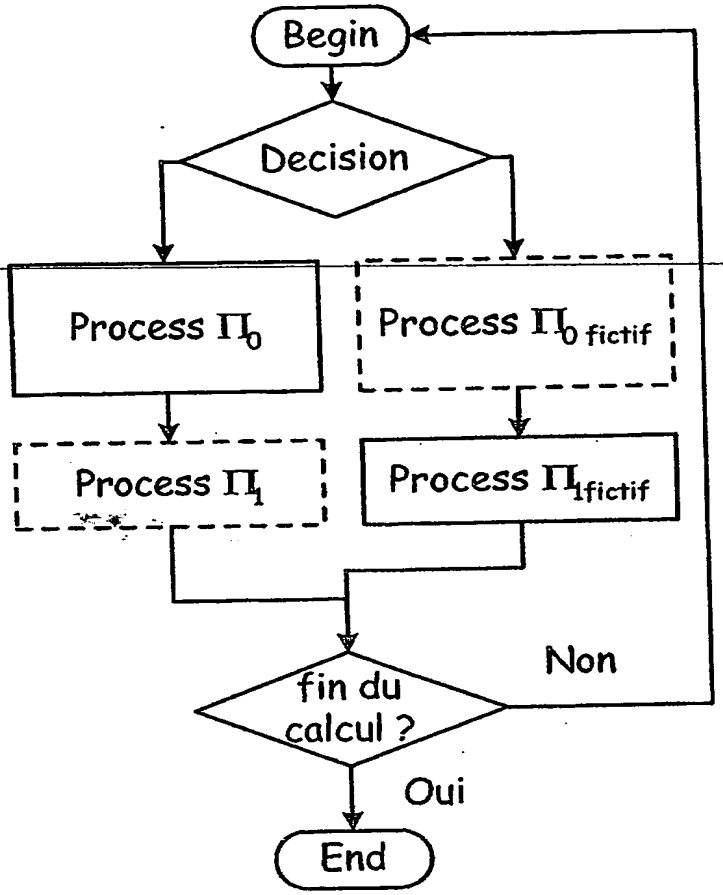


Fig. 1

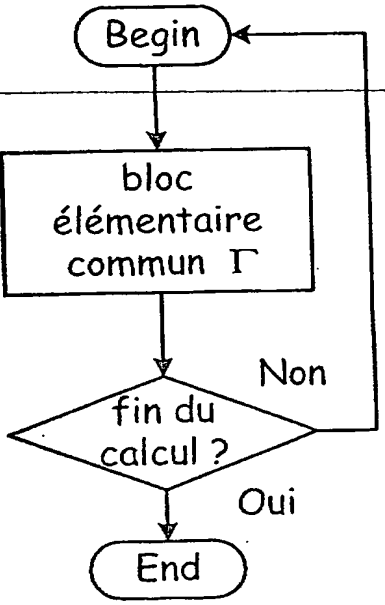


Fig. 2

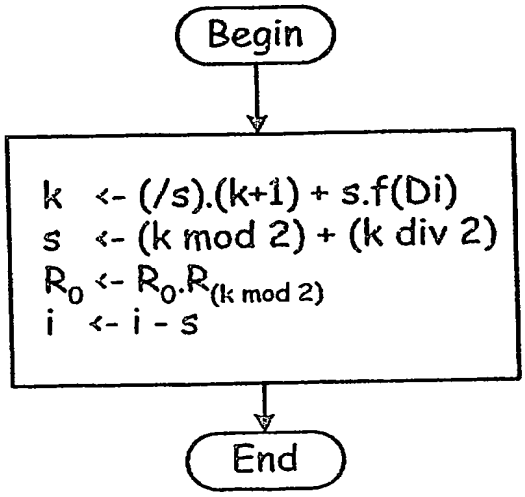


Fig. 3

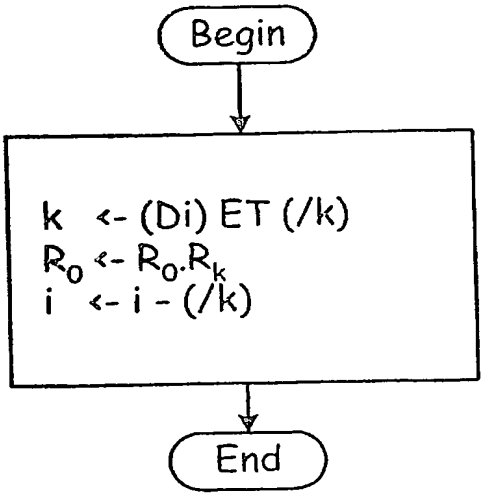


Fig. 4

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa N° 11 235*02

DÉSIGNATION D'INVENTEUR(S) Page N° 1../1..
(Si le demandeur n'est pas l'inventeur ou l'unique inventeur)

Cet imprimé est à remplir lisiblement à l'encre noire DB 113 W /260899

Vos références pour ce dossier *(facultatif)*	016414
N° D'ENREGISTREMENT NATIONAL	0204117

TITRE DE L'INVENTION (200 caractères ou espaces maximum)

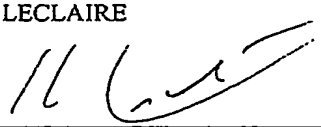
Procédé cryptographique protégé contre les attaques de type à canal caché.

LE(S) DEMANDEUR(S) :
GEMPLUS
Avenue du Pic de Bertagne
Parc d'Activités de GEMENOS
13420 GEMENOS

DESIGNE(NT) EN TANT QU'INVENTEUR(S) : (Indiquez en haut à droite «Page N° 1/1» S'il y a plus de trois inventeurs, utilisez un formulaire identique et numérotez chaque page en indiquant le nombre total de pages).

Nom		JOYE	
Prénoms		Marc	
Adresse	Rue	19, rue Voltaire	
	Code postal et ville	83640	SAINT-ZACHARIE
Société d'appartenance *(facultatif)*			
Nom		CHEVALIER-MAMES	
Prénoms		Benoît	
Adresse	Rue	Résidence Le Général 14, boulevard Ganteaune	
	Code postal et ville	13400	AUBAGNE
Société d'appartenance *(facultatif)*			
Nom			
Prénoms			
Adresse	Rue		
	Code postal et ville		
Société d'appartenance *(facultatif)*			

DATE ET SIGNATURE(S)
DU (DES) DEMANDEUR(S)
OU DU MANDATAIRE
(Nom et qualité du signataire)
Jean Louis LECLAIRE
93.4009



La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire.
Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.